Uniformization Problems for Synchronizations of Automatic Relations on Words

Sarah Winter
RWTH Aachen University, Germany
winter@cs.rwth-aachen.de

Abstract

A uniformization of a binary relation is a function that is contained in the relation and has the same domain as the relation. The synthesis problem asks for effective uniformization for classes of relations and functions that can be implemented in a specific way.

We consider the synthesis problem for automatic relations over finite words (also called regular or synchronized rational relations) by functions implemented by specific classes of sequential transducers.

It is known that the problem "Given an automatic relation, does it have a uniformization by a subsequential transducer?" is decidable in the two variants where the uniformization can either be implemented by an arbitrary subsequential transducer or it has to be implemented by a synchronous transducer. We introduce a new variant of this problem in which the allowed input/output behavior of the subsequential transducer is specified by a set of synchronizations and prove decidability for a specific class of synchronizations.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases automatic relation, uniformization, synchronization, transducer

Digital Object Identifier 10.4230/LIPIcs.ICALP.2018.142

Related Version A full version of the paper is available at `http://arxiv.org/abs/1805.02444`.

Funding Supported by the project (LO 1174/3-1) of the German Research Foundation (DFG).

Acknowledgements The author would like to thank her supervisor Christof Löding for suggesting this topic and his helpful comments and thank the anonymous reviewers of this and an earlier version of the paper for their feedback which greatly improved the presentation.

1 Introduction

A uniformization of a binary relation is a function that selects for each element in the domain of the relation a unique image that is in relation with this element. Of interest to us in this paper are uniformization problems in the setting where the relations and functions on words are defined by finite automata. Relations on words defined by finite automata extend languages defined by finite automata. Unlike for words, different finite automaton models for relations lead to different classes of relations.

Relations defined by asynchronous finite automata are referred to as rational relations. An asynchronous finite automaton is a nondeterministic finite automaton with two tapes whose reading heads can move at different speeds. An equivalent computation model are asynchronous finite transducers (see, e.g., [1]), that is, nondeterministic finite automata whose transitions are labeled by pairs of words.

© Sarah Winter;
licensed under Creative Commons License CC-BY

A well known subclass of rational relations are synchronized rational relations (see [8]), which are defined by synchronous finite automata, that is, finite automata with two tapes whose reading heads move at the same speed. Equivalently, we speak of definability by synchronous finite transducers. The class of synchronized rational relations is also called automatic or regular, here, we use the term automatic.

One uniformization problem asks for proving that each relation in a given class has a certain kind of uniformization. For example, each rational relation can be uniformized by an unambiguous rational function (see [13]). Here, we are interested in the decision version of the problem: Given a relation from some class, does it have a uniformization in some other class? For the class of uniformizations we consider sequential transducers. A sequential transducer reads the input word in a deterministic manner and produces a unique output word for each input word.

The sequential uniformization problem relates to the synthesis problem, which asks, given a specification that relates possible inputs to allowed outputs, whether there is a program implementing the specification, and if so, construct one. This setting originates from Church's synthesis problem [4], where logical specifications over infinite words are considered. Büchi and Landweber [2] showed that for specifications in monadic second order logic, that is, specifications that can be translated into synchronous finite automata, it is decidable whether it can be realized by a synchronous sequential transducer (see, e.g., [14] for a modern presentation of this result). Later, decidability has been extended to asynchronous sequential transducers [10, 9].

Going from the setting of infinite words to finite words uniformization by subsequential [1] transducers is considered. The problem whether a relation given by a synchronous finite automaton can be realized by a synchronous subsequential transducer is decidable; this result can be obtained by adapting the proof from the infinite setting. Decidability has been extended to subsequential transducers [3]. Furthermore, for classes of asynchronous finite automata decidability results for synthesis of subsequential transducers have been obtained in [7].

A semi-algorithm in this spirit was introduced by [11], the algorithm is tasked to synthesize a subsequential transducer that selects the length lexicographical minimal output word for each input word from a given rational relation.

The decision problems that have been studied so far either ask for uniformization by a synchronous subsequential or by an arbitrary subsequential transducer. Our aim is to study the decision problem: Given a rational relation, does it have a uniformization by a subsequential transducer in which the allowed input/output behavior is specified by a given language of synchronizations? The idea is to represent a pair of words by a single word where each position is annotated over $\{1, 2\}$ indicating whether it came from the input or output component. The annotated string provides a synchronization of the pair. It is known that the class of rational relations is synchronized by regular languages [12]. More recently, main subclasses of rational relations have been characterized by their synchronizations [6].

We show decidability for a given automatic relation and a given set of synchronizations that synchronizes an automatic relation. Thus our decidability result generalizes the previously known decidability result for synthesis of synchronous subsequential transducers from automatic relations.

[1] A subsequential transducer can make a final output depending on the last state reached in a run whereas a sequential transducer can only produce output on its transitions.

The paper is structured as follows. First, in Sec. 2, we fix our notations and recap characterizations of synchronization languages established in [6]. In Sec. 3, we introduce uniformization problems with respect to synchronization languages and compare our setting with known results. In Sec. 4, we prove decidability of the question whether an automatic relation has a uniformization by a subsequential transducer in which the input/output behavior is specified by a set of synchronizations that synchronizes an automatic relation.

A full version of this paper can be found online.

2 Synchronizations of relations

Let $\mathbb{N}$ denote the set of all non-negative integers $\{0,1,\dots\}$, and for every $k \in \mathbb{N}$, let $\mathbf{k}$ denote the set $\{1,\dots,k\}$. Given a finite set A, let $|A|$ denote its cardinality and 2^A its powerset.

Languages and relations of finite words. An *alphabet* Σ is a finite set of letters, a finite *word* is a finite sequence over Σ. The set of all finite words is denoted by Σ^* and the empty word by ε. The length of a word $w \in \Sigma^*$ is denoted by $|w|$, the number of occurrences of a letter $a \in \Sigma$ in w by $\#_a(w)$. Given $w \in \Sigma^*$, $w[i]$ stands for the ith letter of w, and $w[i,j]$ for the subword $w[i]\dots w[j]$.

A *language* L over Σ is a subset of Σ^*, and *Pref*(L) is the set $\{u \in \Sigma^* \mid \exists v : uv \in L\}$ of its prefixes. The prefix relation is denoted by $\sqsubseteq$. A *relation* R over Σ is a subset of $\Sigma^* \times \Sigma^*$. The *domain* of a relation R is the set $\mathrm{dom}(R) = \{u \mid (u,v) \in R\}$, the *image* of a relation R is the set $\mathrm{img}(R) = \{v \mid (u,v) \in R\}$. For $u \in \Sigma^*$, let $R(u) = \{v \mid (u,v) \in R\}$ and write $R(u) = v$, if $R(u)$ is a singleton.

A *regular expression* r over Σ has the form $\emptyset$, ε, $\sigma \in \Sigma$, $r_1 \cdot r_2$, $r_1 + r_2$, or r_1^* for regular expressions r_1, r_2. The term r^+ is short for $r \cdot r^*$. The concatenation operator $\cdot$ is often omitted. The language associated to r is defined as usual, denoted $L(r)$, or conveniently, r.

▶ **Definition 1** (synchronization, L-controlled [6])**.** For $c \in \{\mathtt{i},\mathtt{o}\}$, referring to input and output, respectively, we define two morphisms $\pi_c\colon (\mathbf{2} \times \Sigma) \to \Sigma \cup \{\varepsilon\}$ by $\pi_{\mathtt{i}}((i,a)) = a$ if $i = 1$, otherwise $\pi_{\mathtt{i}}((i,a)) = \varepsilon$, and likewise for $\pi_{\mathtt{o}}$ with $i = 2$. These morphisms are lifted to words over $(\mathbf{2} \times \Sigma)$.

A word $w \in (\mathbf{2} \times \Sigma)^*$ is a *synchronization* of a uniquely determined pair (w_1, w_2) of words over Σ, where $w_1 = \pi_{\mathtt{i}}(w)$ and $w_2 = \pi_{\mathtt{o}}(w)$. We write $[\![w]\!]$ to denote (w_1,w_2). Naturally, a set $S \subseteq (\mathbf{2} \times \Sigma)^*$ of synchronizations defines the relation $[\![S]\!] = \{[\![w]\!] \mid w \in S\}$.

A word $w = (i_1,a_1)\dots(i_n,a_n) \in (\mathbf{2} \times \Sigma)^*$ is the convolution $u \otimes v$ of two words $u = i_1 \dots i_n \in \mathbf{2}^*$ and $v = a_1 \dots a_n \in \Sigma^*$. Given a language $L \subseteq \mathbf{2}^*$, we say w is *L-controlled* if $u \in L$. A language $S \subseteq (\mathbf{2} \times \Sigma)^*$ is *L-controlled* if all its words are.

A language $L \subseteq \mathbf{2}^*$ is called a *synchronization language*. For a regular language $L \subseteq \mathbf{2}^*$, $\textsc{Rel}(L) = \{[\![S]\!] \mid S \text{ is a regular } L\text{-controlled language}\}$ is the set of relations that can be given by L-controlled synchronizations. Let $\mathcal{C}$ be a class of relations, we say L *synchronizes* $\mathcal{C}$ if $\textsc{Rel}(L) \subseteq \mathcal{C}$.

▶ **Definition 2** (lag, shift, shiftlag [6])**.** Given a word $w \in \mathbf{2}^*$, a position $i \leq |w|$, and $\gamma \in \mathbb{N}$. We say i is *γ-lagged* if $|\#_1(w[1,i]) - \#_2(w[1,i])| = \gamma$, and likewise, we define *$>\gamma$-lagged* and *$<\gamma$-lagged*. A *shift* of w is a position $i \in \{1,\dots,|w|-1\}$ such that $w[i] \neq w[i+1]$. Two shifts $i < j$ are *consecutive* if there is no shift l such that $i < l < j$. Let *shift*(w) be the number of shifts in w, let *lag*(w) be the maximum lag of a position in w, and let *shiftlag*(w) be the maximum $n \in \mathbb{N}$ such that w contains n consecutive shifts which are $>n$-lagged.

We lift these notions to languages by taking the supremum in $\mathbb{N} \cup \{\infty\}$, e.g., *shift*$(L) = \sup\{\textit{shift}(w) \mid w \in L\}$, and likewise for *lag*$(L)$ and *shiftlag*(L).

The following characterizations for well known subclasses of rational relations were shown in [6]. Recall, rational relations are definable by asynchronous finite automata, automatic relations by synchronous finite automata, and recognizable relations are definable as finite unions of products of regular languages. We omit a formal definition of these models since it is not relevant to this paper.

▶ **Theorem 3** ([6]). *Let $L \subseteq \mathbf{2}^*$ be a regular language. Then:*
1. *L synchronizes recognizable relations iff $\mathit{shift}(L) < \infty$,*
2. *L synchronizes automatic relations iff $\mathit{shiftlag}(L) < \infty$,*
3. *L synchronizes rational relations.*

For ease of presentation, let $\Sigma_{\mathtt{io}}$, $\Sigma_{\mathtt{i}}$, and $\Sigma_{\mathtt{o}}$ be short for $\mathbf{2} \times \Sigma$, $\{1\} \times \Sigma$, and $\{2\} \times \Sigma$, respectively. If convenient, we use distinct symbols for input and output, instead of symbols annotated with 1 or 2.

For the results shown in this paper, it is useful to lift some notions introduced in [6] from words and languages over $\mathbf{2}$ to words and languages over $\Sigma_{\mathtt{io}}$.

▶ **Definition 4.** We lift the notions of *lag*, *shift*, and *shiftlag* from words and languages over $\mathbf{2}$ to words and languages over $\Sigma_{\mathtt{io}}$ in the natural way.

Furthermore, given a language $T \subseteq \Sigma_{\mathtt{io}}^*$, we say a word $w \in \Sigma_{\mathtt{io}}^*$ is T-controlled if $w \in T$. A language $S \subseteq \Sigma_{\mathtt{io}}^*$ is T-controlled if all its words are, namely, if $S \subseteq T$.

Automata on finite words. We fix our notations concerning finite automata on finite words. A *nondeterministic finite automaton* (*NFA*) is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$, where Q is a finite set of states, Σ is a finite alphabet, $q_0 \in Q$ is the initial state, $\Delta \subseteq Q \times \Sigma \times Q$ is the transition relation, and $F \subseteq Q$ is the set of final states. A *run* ρ of $\mathcal{A}$ on $w = a_1 \dots a_n \in \Sigma^*$ is a sequence of states $p_0 p_1 \dots p_n$ such that $(p_i, a_{i+1}, p_{i+1}) \in \Delta$ for all $i \in \{0, \dots, n-1\}$. Shorthand, we write $\mathcal{A} : p_0 \xrightarrow{w} p_n$. A run is *accepting* if it starts in q_0 and ends in a state from F. The language *recognized* by $\mathcal{A}$, written $L(\mathcal{A})$, is the set of words $w \in \Sigma^*$ that admit an accepting run of $\mathcal{A}$ on w. For $q \in Q$, let $\mathcal{A}_q$ denote the NFA obtained from $\mathcal{A}$ by setting its initial state to q. The class of languages recognized by NFAs is the class of regular languages. An NFA is *deterministic* (a *DFA*) if for each state $q \in Q$ and $a \in \Sigma$ there is at most one outgoing transition. In this case, it is more convenient to express Δ as a (partial) function $\delta : Q \times \Sigma \to Q$. Furthermore, let δ^* denote the usual extension of δ from letters to words.

We introduce some notions only applicable if an NFA recognizes a set of synchronizations. Given a regular $S \subseteq \Sigma_{\mathtt{io}}^*$, let $\mathcal{A} = (Q, \Sigma_{\mathtt{io}}, q_0, \Delta, F)$ be an NFA that recognizes S. We define $Q^{\mathtt{i}} = \{p \in Q \mid \exists a \in \Sigma, q \in Q : (p,(1,a),q) \in \Delta\}$ and $Q^{\mathtt{o}} = \{p \in Q \mid \exists a \in \Sigma, q \in Q : (p,(2,a),q) \in \Delta\}$ as the sets of states that have outgoing transitions from which input and output can be consumed, respectively. If ($Q^{\mathtt{i}}$,$Q^{\mathtt{o}}$) is a partition of Q, we write $Q = Q^{\mathtt{i}} \uplus Q^{\mathtt{o}}$. We call $\mathcal{A}$ *sequential* if $\mathcal{A}$ is deterministic, and $Q = Q^{\mathtt{i}} \uplus Q^{\mathtt{o}}$, and each $q \in Q^{\mathtt{o}}$ has at most one outgoing transition. For short, we refer to a sequential DFA as *sDFA*. Finally, we define the input automaton $\mathcal{A}_D$ of $\mathcal{A}$ as $(Q, \Sigma, q_0, \Delta', F)$, where $\Delta' = \{(p,a,q) \mid \mathcal{A} : p \xrightarrow{w} q \text{ and } \pi_{\mathtt{i}}(w) = a \in \Sigma\}$. A comparison to standard transducer models is given in the next section.

3 Uniformization problems

A *uniformization* of a relation $R \subseteq \Sigma^* \times \Sigma^*$ is a complete function $f_R : \mathrm{dom}(R) \to \Sigma^*$ with $(u, f_R(u)) \in R$ for all $u \in \mathrm{dom}(R)$. If such a function is given as a relation R_f, we write $R_f \subseteq_{\mathsf{u}} R$ to indicate that R_f is a uniformization of R.

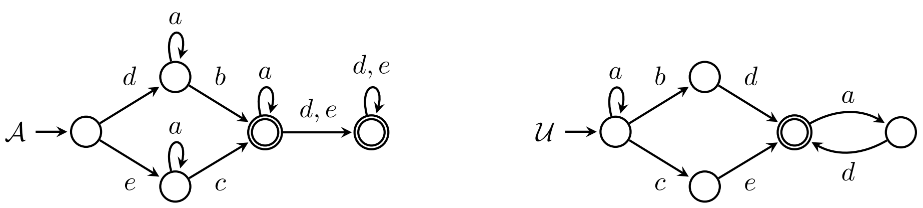


Figure 1 Cf. Ex.6; $S = L(\mathcal{A})$ and $U = L(\mathcal{U})$, we have $[\![U]\!] \subseteq_{\mathsf{u}} [\![S]\!]$.

▶ **Definition 5** (Resynchronized uniformization problem). The *resynchronized uniformization problem* asks, given a regular source language $S \subseteq \Sigma_{\mathtt{io}}^*$ and a regular target language $T \subseteq \Sigma_{\mathtt{io}}^*$, whether there exists a regular language $U \subseteq T$ recognized by a sequential DFA such that $[\![U]\!] \subseteq_{\mathsf{u}} [\![S]\!]$.

▶ **Example 6.** Let $\Sigma_{\mathtt{i}} = \{a, b, c\}$ and $\Sigma_{\mathtt{o}} = \{d, e\}$, let $S \subseteq \Sigma_{\mathtt{io}}^*$ be given by $\mathcal{A}$ depicted in Fig. 1. The recognized relation is $[\![S]\!] = \{(a^iba^j, d(d+e)^k) \mid i, j, k \geq 0\} \cup \{(a^ica^j, e(d+e)^k) \mid i, j, k \geq 0\}$. Furthermore, let $T = \Sigma_{\mathtt{i}}^*(\Sigma_{\mathtt{i}}\Sigma_{\mathtt{o}})^+$. A T-controlled uniformization U is given by the sequential DFA $\mathcal{U}$ depicted in Fig. 1. The recognized relation is $[\![U]\!] = \{(a^iba^j, dd^j) \mid i, j, k \geq 0\} \cup \{(a^ica^j, ed^j) \mid i, j \geq 0\}$.

Comparing our definition of sequential DFAs with standard transducer models we notice that sequential transducers directly correspond to sequential DFAs. See, e.g., [1] for an introduction to transducers. Our model can be modified to correspond to subsequential transducers (which can make a final output after the word has ended) by slightly modifying the representation of the relation by adding a dedicated endmarker in the usual way.

In the remainder it is implicitly assumed that every given source and target language is represented with endmarkers, thus our stated results correspond to uniformization by subsequential transducers.

Our main result is the decidability of the resynchronized uniformization problem for a given automatic relation and a given set of synchronizations controlled by a language that synchronizes automatic relations. In Sec. 4 we see that our decidability result is obtained by a reduction to the following simpler uniformization problem.

▶ **Definition 7** (Subset uniformization problem). The *subset uniformization problem* asks, given a regular language $S \subseteq \Sigma_{\mathtt{io}}^*$, whether there exists a regular language $U \subseteq S$ recognized by a sequential DFA such that $[\![U]\!] \subseteq_{\mathsf{u}} [\![S]\!]$.

The notion of subset uniformization directly corresponds to the notion of sequential $\mathbb{I}$-uniformization introduced in [7]. It was shown that deciding the sequential $\mathbb{I}$-uniformization problem reduces to deciding which player has a winning strategy in a safety game between In and Out. Hence, we directly obtain the following result.

▶ **Theorem 8** ([7]). *The subset uniformization problem is decidable.*

Now that we have formulated our uniformization problems, we link these to known uniformization problems. Asking whether a relation has a $\Sigma_{\mathtt{io}}^*$-controlled subsequential uniformization is equivalent to asking whether it has a uniformization by an arbitrary subsequential transducer. Asking whether a relation has a $(\Sigma_{\mathtt{i}}\Sigma_{\mathtt{o}})^*(\Sigma_{\mathtt{i}}^* + \Sigma_{\mathtt{o}}^*)$- resp. $\Sigma_{\mathtt{i}}^*\Sigma_{\mathtt{o}}^*$-controlled subsequential uniformization is equivalent to asking whether it has a uniformization by a synchronous subsequential transducer resp. by a transducer that reads the complete input before producing output.

Table 1 Overview over decidability results. The columns list the type of relation to be uniformized. The rows list the type of synchronization used as uniformization parameter; the upper three rows list fixed languages of synchronizations, the lower three rows list parameter classes, where 'rational' means the given set of allowed synchronizations is controlled by an arbitrary synchronization language, 'automatic' (resp. 'recognizable') means the given set of allowed synchronizations is controlled by a synchronization language that synchronizes automatic (resp. recognizable) relations.

relation / sync.	rational	deterministic rational	finite-valued	automatic	recog-nizable
$\Sigma_{\mathtt{io}}^*$	undec. [3]	dec. [7]	dec. [7]	dec. [3]	dec.
$(\Sigma_{\mathtt{i}}\Sigma_{\mathtt{o}})^*(\Sigma_{\mathtt{i}}^* + \Sigma_{\mathtt{o}}^*)$	undec. [3]	?	?	dec. [2]	dec.
$\Sigma_{\mathtt{i}}^*\Sigma_{\mathtt{o}}^*$	?	?	?	dec. [3]	dec.
rational	undec.	?	?	?	dec.
automatic	undec.	?	?	**dec.**	dec.
recognizable	?	?	?	dec.	dec.

Table 1 provides an overview over known and new decidability results of the resynchronized uniformization problem for different types of relations and synchronization parameters. Our main result is the decidability for a given automatic relation and a given set of allowed synchronizations that is controlled by a synchronization language that synchronizes automatic relations. The decidability results in the rightmost column can be shown by a simple reduction to the subset uniformization problem which is presented in the full version of the paper. The other entries in the lower three rows are simple consequences of the results presented in the upper three rows resp. our main result.

Regarding the table entry where the relation is automatic and a desired uniformizer is $(\Sigma_{\mathtt{i}}\Sigma_{\mathtt{o}})^*(\Sigma_{\mathtt{i}}^* + \Sigma_{\mathtt{o}}^*)$-controlled, there is an alternative formulation of the decision problem in the case that the given relation is $(\Sigma_{\mathtt{i}}\Sigma_{\mathtt{o}})^*(\Sigma_{\mathtt{i}}^* + \Sigma_{\mathtt{o}}^*)$-controlled (the usual presentation for automatic relations, e.g., by a synchronous transducer). In this case the problem can also be stated as the question whether the relation has a subset uniformization.

We now generalize this to Parikh-injective synchronization languages. Given some $L \subseteq \mathbf{2}^*$, let $\Pi_L : L \to \mathbb{N}^2$ be the function that maps a word $w \in L$ to its *Parikh image*, that is to the vector $(\#_1(w), \#_2(w))$. We say L is *Parikh-injective* if Π_L is injective.

▶ **Proposition 9.** Let $L \subseteq \mathbf{2}^*$ be a regular Parikh-injective language, let $S \subseteq \Sigma_{\mathtt{io}}^*$ be an L-controlled regular language and let $T = \{w \in \Sigma^* \mid w \text{ is } L\text{-controlled}\}$. Every T-controlled uniformization of S is a subset uniformization of S.

Given L, S and T as in Proposition 9, it directly follows that the resynchronized uniformization problem is equivalent to the subset uniformization problem, which is decidable by Theorem 8.

4 Automatic uniformizations of automatic relations

Here we present our main result stating that it is decidable whether a given automatic relation has a uniformization by a subsequential transducer whose induced set of synchronizations is controlled by a given regular language that synchronizes automatic relations.

▶ **Theorem 10.** *Given a regular source language with finite shiftlag and a regular target language with finite shiftlag. Then, the resynchronized uniformization problem is decidable.*

In [6], it is shown that $(12)^*(1^*+2^*)$ is an effective canonical representative of the class RL_{FSL} of regular languages with finite *shiftlag*. Meaning that for every $L \in \mathsf{RL}_{FSL}$ and every $R \in \textsc{Rel}(L)$, there is an effectively constructible $(12)^*(1^*+2^*)$-controlled regular language S so that $[\![S]\!] = R$.

In the remainder of this section, let $S \subseteq \Sigma_{\mathtt{io}}^*$ be a regular source language with finite *shiftlag*. Also, let S_{can} be the equivalent $(12)^*(1^*+2^*)$-controlled language with $[\![S_{can}]\!] = [\![S]\!]$. Furthermore, let $T \subseteq \Sigma_{\mathtt{io}}^*$ be a regular target language with finite *shiftlag*.

▶ **Assumption 11.** We assume that S_{can} is recognized by a DFA $\mathcal{A} = (Q_\mathcal{A}, \Sigma_{\mathtt{io}}, q_0^\mathcal{A}, \Delta_\mathcal{A}, F_\mathcal{A})$, T is recognized by a DFA $\mathcal{B} = (Q_\mathcal{B}, \Sigma_{\mathtt{io}}, q_0^\mathcal{B}, \Delta_\mathcal{B}, F_\mathcal{B})$ and $shiftlag(T) < n$.

For notational convenience, given $x \in \Sigma_{\mathtt{i}}^*$ and $y \in \Sigma_{\mathtt{o}}^*$, we write $\delta_\mathcal{A}^*(q,(x,y))$ to mean $\delta_\mathcal{A}^*(q,w)$, where $w \in \Sigma_{\mathtt{io}}$ is the canonical synchronization of x and y, i.e., w is the $(12)^*(1^*+2^*)$-controlled synchronization of the pair (x,y).

The remainder of this section is devoted to the proof of Theorem 10. The proof is split in two main parts; the goal of the first part is to show that if S has a T-controlled uniformization by an sDFA, then S has a T_k-controlled uniformization by an sDFA for a regular $T_k \subseteq T$ that is less complex than T, cf. Lemma 23. The goal of the second part is to show that the set $T_k(S)$ defined by $\{w \mid w \in T_k \text{ and } [\![w]\!] \in [\![S]\!]\}$ is regular and computable (due to the form of T_k), cf. Lemma 24. Then, to conclude the proof, we show that the question whether S has a T-controlled uniformization by an sDFA can be reduced to the question whether $T_k(S)$ has a subset uniformization by an sDFA, which is decidable by Theorem 8.

Towards giving an exact description of T_k, consider the following auxiliary lemma characterizing the form of regular synchronization languages with finite *shiftlag*. Given $\nu \in \mathbb{N}$, we denote by $L_{\leq\nu}$ the regular set of words over $\mathbf{2}$ with $\leq\nu$-lagged positions, i.e., $L_{\leq\nu} = \{u \in \mathbf{2}^* \mid lag(u) \leq \nu\}$; we denote by $T_{\leq\nu}$ the regular set of words over $\Sigma_{\mathtt{io}}$ with $\leq\nu$-lagged positions, i.e., $T_{\leq\nu} = \{w \in \Sigma_{\mathtt{io}}^* \mid lag(w) \leq \nu\}$.

▶ **Lemma 12** ([6]). *Given a regular language $L \subseteq \mathbf{2}^*$ with $shiftlag(L) < m$. It holds that $L \subseteq L_{\leq\nu} \cdot (1^*+2^*)^m$ with ν chosen as $2\,(m(|Q|+1)+1)$, where Q is the state set of an NFA recognizing L.*

Clearly, this lemma can be lifted to regular languages over $\Sigma_{\mathtt{io}}$. Based on Asm. 11 and Lemma 12, we can make the following assumption.

▶ **Assumption 13.** Assume that $T \subseteq T_{\leq\gamma} \cdot (\Sigma_{\mathtt{i}}^* + \Sigma_{\mathtt{o}}^*)^n$ with $\gamma = 2\,(n(|Q_\mathcal{B}|+1)+1)$.

Now, we can be more specific about $T_k \subseteq T$.

▶ **Definition 14.** For $i \geq 0$, let T_i be the set $T \cap \left(T_{\leq\gamma} \cdot (\Sigma_{\mathtt{i}}^* + \Sigma_{\mathtt{o}}^{\leq i})^n\right)$, that is, the set of $w \in T$ such that after a position in w is more than γ-lagged, the number of output symbols per block is at most i.

Our aim is to show that there is a bound k such that S has either a T_k-controlled uniformization by an sDFA or no T-controlled uniformization by an sDFA. From now on, we call an sDFA implementing a uniformization simply a uniformizer.

The main difficulty in solving the resynchronized uniformization problem is that in general a uniformizer can have unbounded lag, because the waiting time between shifts can be arbitrarily long. The key insight for the proof is that if such a long waiting time for a shift from input to output is necessary, then, in order to determine the next output block, it is not necessary to store the complete input that is ahead. We show that it suffices to consider an abstraction of the input that is ahead. Therefore we will introduce input profiles based on state transformation trees we define below.

Similarly, to deal with the situation where there is a long waiting time for a shift from output to input, we introduce output profiles as an abstraction of output that is ahead.

The bound on the length of output blocks will be chosen based on the profiles. Before defining profiles, we introduce some necessary definitions and notions.

Trees. A *finite unordered unranked tree* over an alphabet, a tree for short, is a finite non-empty directed graph with a distinguished root node, such that for any node, there exists exactly one path from the root to this node. Additionally, a mapping from the nodes of the graph to the alphabet is given. More formally, a *tree* t over Σ is given by a tuple (V_t, E_t, v_t, val_t), where V_t is a non-empty set of nodes, $E_t \subseteq V_t \times V_t$ is a set of edges, v_t is the root of t, also denoted $root(t)$, and val_t is a mapping $V_t \to \Sigma$. Furthermore, it is satisfied that any node is reached by a unique path from the root. Let T_Σ denote the set of all trees over Σ. We only distinguish trees up to isomorphism.

Given a tree t and a node v of t, let $t|_v$ denote the *subtree* of t rooted at v.

An $a \in \Sigma$ can also be seen as a tree $a \in T_\Sigma$ defined by $(\{v\}, \emptyset, v, val_a)$, where $val_a(v) = a$.

For two trees t_1 and t_2 with $val_{t_1}(root(t_1)) = val_{t_2}(root(t_2))$, i.e., with the same root label, we define $t_1 \circ t_2$ as the tree t given by $(V_t, E_t, root(t_1), val_t)$, where $V_t = V_{t_1} \cup V_{t_2} \setminus \{root(t_2)\}$, $E_t = E_{t_1} \cup \{(root(t), v) \mid (root(t_2), v) \in E_{t_2}\} \cup (E_{t_2} \setminus \{(root(t_2), v) \in E_{t_2}\})$ and val_t as $val_{t_1} \cup val_{t_2}$ over nodes in V_t (assuming $V_{t_1} \cap V_{t_2} = \emptyset$).

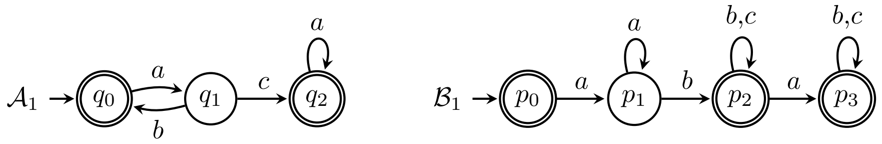
Given $a \in \Sigma$ and trees $t_1, \dots, t_n$, we define $a(t_1 \dots t_n)$ to be the tree (V_t, E_t, r, val_t), where $V_t = \bigcup_{i=1}^n V_{t_i} \cup \{r\}$ with a new node r, $E_t = \bigcup_{i=1}^n E_{t_i} \cup \{(r, root(t_i)) \mid 1 \leq i \leq n\}$ and val_t is defined as $val_t(r) = a$ and $\bigcup_{i=1}^n val_{t_i}$ (assuming $V_{t_i} \cap V_{t_j} = \emptyset$ for all $i \neq j$).

State transformation trees. Now that we have fixed our notations, we explain what kind of information we want to represent using state transformation trees. Basically, for an input segment that is ahead and causes lag, we are interested in how the input segment can be combined with output segments of same or smaller length and how this output can be obtained.

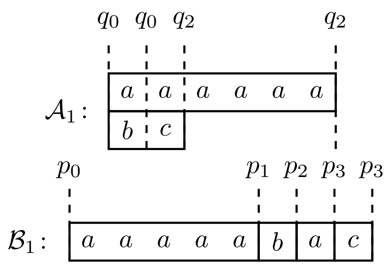
In the following we give an intuitive example.

▶ **Example 15.** Let $\Sigma_{\mathtt{i}} = \{a\}$ and $\Sigma_{\mathtt{o}} = \{b, c\}$. Consider the language $S_1 \subseteq \Sigma_{\mathtt{io}}^*$ given by the DFA $\mathcal{A}_1$ depicted in Fig. 2a, and the language $T_1 \subseteq \Sigma_{\mathtt{io}}^*$ given by the DFA $\mathcal{B}_1$ depicted in Fig. 2a. As we can see, S_1 is $(12)^*(1^* + 2^*)$-controlled, thus, already in its canonical form, and T_1 is $1^*2^*1^*2^*$-controlled. Both languages have finite *shiftlag*.

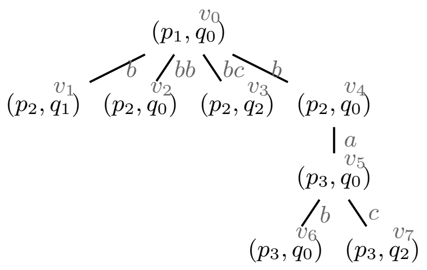
Generally, a T_1-controlled uniformizer of S_1 can have arbitrary large lag. We take a look at the runs starting from q_0 in $\mathcal{A}_1$ and starting from p_0 in $\mathcal{B}_1$ that the computation of such a uniformizer can induce. However, $\mathcal{A}_1$ can only be simulated on the part where the lag is recovered, but arbitrarily large lag can occur, thus our goal is to find an abstraction of the part that causes lag. E.g., assume that such a uniformizer reads aa without producing output. Towards defining an abstraction of aa, we are interested in how aa could be combined with outputs of same or smaller length and how these outputs could be produced by some T_1-controlled uniformizer. Such a uniformizer could read some more as and eventually must produce output. Reading as leads from p_0 to p_1 in $\mathcal{B}_1$. There are a few possibilities how output of length at most two can be produced such that it is valid from p_1 and the simulation from q_0 can be continued. It is possible to output b ($\delta^*_{\mathcal{B}_1}(p_1, b) = p_2$, $\delta^*_{\mathcal{A}_1}(q_0, aba) = q_1$), bb ($\delta^*_{\mathcal{B}_1}(p_1, bb) = p_2$, $\delta^*_{\mathcal{A}_1}(q_0, abab) = q_0$) or bc ($\delta^*_{\mathcal{B}_1}(p_1, bc) = p_2$, $\delta^*_{\mathcal{A}_1}(q_0, abac) = q_2$). Alternatively, it is possible to output b ($\delta^*_{\mathcal{B}_1}(p_1, b) = p_2$, $\delta^*_{\mathcal{A}_1}(q_0, ab) = q_0$) read another a ($\delta^*_{\mathcal{B}_1}(p_2, a) = p_3$) and then produce b ($\delta^*_{\mathcal{B}_1}(p_3, b) = p_3$, $\delta^*_{\mathcal{A}_1}(q_0, ab) = q_0$) or c ($\delta^*_{\mathcal{B}_1}(p_3, c) = p_3$, $\delta^*_{\mathcal{A}_1}(q_0, ac) = q_2$). We see that the outputs bb and bc can each be obtained in two different ways. Namely, as one single output block, or as two output blocks with an input block in



(a) $\Sigma_{\mathtt{i}} = \{a\}$, $\Sigma_{\mathtt{o}} = \{b, c\}$. $\mathcal{A}_1$ recognizes S_1, $\mathcal{B}_1$ recognizes T_1. S_1 is $(12)^*(1^* + 2^*)$-controlled and T_1 is $1^*2^*1^*2^*$-controlled, thus both have finite *shiftlag*. S_1 is already in the canonical form.



(b) Runs of $\mathcal{A}_1$ and $\mathcal{B}_1$ on synchronizations of $(aaaaaa, bc)$. $\mathcal{A}_1$ runs on the canonical synchronization, i.e., on $abacaaaa$. To illustrate this, input and output are drawn one above the other.



(c) $\text{STT}^1(aa, p_1, q_0)$. The combination of both runs shown in Fig. 2b is reflected by the rightmost path in the state transformation tree.

Figure 2 A source language S_1 and a target language T_1 are given in Fig. 2a. A pair and two different synchronizations of said pair as well as runs are given in Fig. 2b. The state transformation tree $\text{STT}^1(aa, p_1, q_0)$ is given in Fig. 2c, its edges are labeled with the respective associated words and its vertices are named for easier reference in Ex. 15. For a formal definition of STTs see Def. 16, for an explanation for this specific tree see Ex. 15.

between (w.r.t. $\mathcal{B}_1$, we do not care about the number of blocks w.r.t. $\mathcal{A}_1$). The maximal number of considered output blocks (w.r.t. the target synchronization) is parameterized in the formal definition.

We take a look at the tree in Fig. 2c, this tree contains all the state transformations that can be induced by the described possibilities. The possibilities to produce output in one single block is reflected by the edges (v_0, v_1), (v_0, v_2) and (v_0, v_3) representing the state transformation induced by the respective output block. The possibilities to produce output in two blocks is reflected by the edges (v_0, v_4) representing the state transformation induced by the first output block, (v_4, v_5) representing the state transformation induced by the intermediate input block, (v_5, v_6) and (v_5, v_7) representing the state transformation induced by the respective second output block.

Now that we have given some intuition, we formally introduce input state transformation trees, a graphical representation of the construction of input state transformation trees is given in Fig. 3. As seen in the example, each edge of such a tree represents the state transformation induced by an output resp. input block, alternatively.

▶ **Definition 16** (Input state transformation tree). For $i \geq 0$, $p \in Q_{\mathcal{B}}$, $q \in Q_{\mathcal{A}}$ and $x \in \Sigma_{\mathtt{i}}^*$, the *state transformation tree* $\text{STT}^i(x, p, q)$ is a tree over $Q_{\mathcal{B}} \times Q_{\mathcal{A}}$ defined inductively.

- For $i = 0$, the tree $\text{STT}^0(x, p, q)$ is built up as follows.
  Let $\text{Reach}_0 \subseteq Q_{\mathcal{B}} \times Q_{\mathcal{A}}$ be the smallest set such that $(p', q') \in \text{Reach}_0$ if there is some $y \in \Sigma_{\mathtt{o}}^*$ with $|y| \leq |x|$ such that $\delta_{\mathcal{A}}^*(q, (x, y)) = q'$ and $\delta_{\mathcal{B}}^*(p, y) = p'$.
  (This set represents state transformations induced by output blocks that fully consume x.)
  Then the tree $\text{STT}^0(x, p, q)$ is defined as $(p, q)(r_1 \ldots r_n)$ for $\text{Reach}_0 = \{r_1, \ldots, r_n\}$, meaning it contains a child for every state transformation that can be induced w.r.t. $\mathcal{A}$ and $\mathcal{B}$ starting from q and p, respectively, by the input segment x together with an output segment that consumes x (w.r.t. $\mathcal{A}$) consisting of a single output block (w.r.t. $\mathcal{B}$).

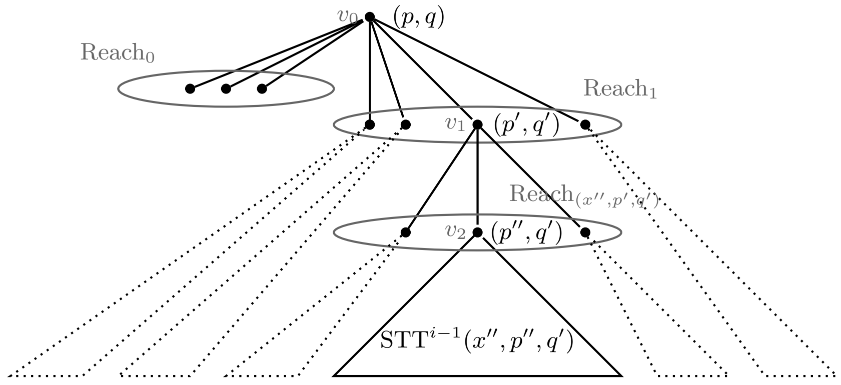


Figure 3 Schema of the input state transformation tree $\mathrm{STT}^i(x,p,q)$ for some $i>0$. Cf. Def. 16. Let $x'x''$ be a factorization of x with $x',x''\in\Sigma_{\mathtt{i}}^+$, and let $y\in\Sigma_{\mathtt{o}}^+$ be such that $|x'|=|y|$ and $\delta_{\mathcal{A}}^*(q,(x',y))=q'$ and $\delta_{\mathcal{B}}^*(p,y)=p'$, and let $\delta_{\mathcal{B}}^*(p',w)=p''$ for some $w\in\Sigma_{\mathtt{i}}^+$, then $\mathrm{STT}^i(x,p,q)$ contains a path $v_0v_1v_2$ labeled $(p,q)(p',q')(p'',q')$ such that v_0 is the root, v_1 is the root of $t^{i-1}_{(x'',p',q')}$, and v_2 is the root of $\mathrm{STT}^{i-1}(x'',p'',q')$.

- For $i>0$, the tree $\mathrm{STT}^i(x,p,q)$ is built up as follows.
  Let $\mathrm{Reach}_1\subseteq\Sigma_{\mathtt{i}}^*\times Q_{\mathcal{B}}\times Q_{\mathcal{A}}$ be the smallest set such that $(x'',p',q')\in\mathrm{Reach}_1$ if
  - $x=x'x''$ with $x''\in\Sigma_{\mathtt{i}}^+$ for an $x'\in\Sigma_{\mathtt{i}}^+$ such that there is a $y\in\Sigma_{\mathtt{o}}^+$ with $|y|=|x'|$, and
  - $\delta_{\mathcal{A}}^*(q,(x',y))=q'$ and $\delta_{\mathcal{B}}^*(p,y)=p'$.

  (This set represents state transformations induced by output blocks that partially consume x.)
  For $(x'',p',q')\in\mathrm{Reach}_1$, let $\mathrm{Reach}_{(x'',p',q')}\subseteq\Sigma_{\mathtt{i}}^*\times Q_{\mathcal{B}}\times Q_{\mathcal{A}}$ be the smallest set such that $(x'',p'',q')\in\mathrm{Reach}_{(x'',p',q')}$ if $\delta_{\mathcal{B}}^*(p',w)=p''$ for some $w\in\Sigma_{\mathtt{i}}^+$.
  (These sets represents state transformations induced by intermediate input blocks.)
  Furthermore, let the tree $t^{i-1}_{(x'',p',q')}$ be defined as $(p',q')(\mathrm{STT}^{i-1}r_1\dots\mathrm{STT}^{i-1}r_n)$ for $\mathrm{Reach}_{(x'',p',q')}=\{r_1,\dots,r_n\}$.
  Then the tree $\mathrm{STT}^i(x,p,q)$ is defined as

  $$\mathrm{STT}^0(x,p,q)\circ(p,q)(t^{i-1}_{s_1}\dots t^{i-1}_{s_n})$$

  for $\mathrm{Reach}_1=\{s_1,\dots,s_n\}$, meaning it contains a path for every sequence of state transformations that can be induced w.r.t. $\mathcal{A}$ and $\mathcal{B}$ starting from q and p, respectively, by the input segment x together with an output segment that consumes x (w.r.t. $\mathcal{A}$) consisting of at most $i+1$ output blocks (w.r.t. $\mathcal{B}$). Additionally, for output segments that have a common prefix of output blocks the state transformations induced by the common prefix of blocks are represented by the same nodes in the tree.

Intuitively, edges in such a tree are associated with the words that induced the state transformation, e.g., as shown in Fig 2c.

Given a tree as in Def. 16, the maximal degree of such a tree depends on the input word used as parameter. Our goal is to have state transformation trees where the maximum degree is independent of this parameter. Therefore, we introduce *reduced trees.* The idea is that if for some input word different outputs induce the same state transformations then only one representation is kept in the input state transformation tree.

▶ **Definition 17** (Reduced tree)**.** A tree $t\in T_\Sigma$ over some alphabet Σ is called *reduced* if for each node v there exist no two children u,u' of v such that the subtrees rooted at u and u' are isomorphic.

For a tree $t \in T_\Sigma$, let $\mathit{red}(t) \in T_\Sigma$ denote its reduced variant. The reduced variant of a tree can easily be obtained by a bottom-up computation where for each node duplicate subtrees rooted at its children are removed.

Note that for each i, the set of reduced input state transformation trees with parameter i is a finite set.

Hitherto, we have discussed how to capture state transformations induced by an input word together with output words of same or smaller length. Additionally, we need to capture state transformations induced by an output word together with input words of same or smaller length. Therefore, we introduce a notion similar to input state transformation trees, namely, *output state transformation trees*. A formal definition can be found in the full version.

Furthermore, we need a notion that captures state transformations that can be induced by an input resp. output word alone, see Def. 18 below. Then, we are ready to define profiles.

▶ **Definition 18** (State transformation function)**.** For each $w \in \Sigma_{\mathtt{i}}^* \cup \Sigma_{\mathtt{o}}^*$, we define the function $\tau_w \colon Q_{\mathcal{B}} \to Q_{\mathcal{B}}$ with $\tau_w(p) = q$ if $\delta_{\mathcal{B}}^*(p, w) = q$ called *state transformation function w.r.t. w.*

Profiles. Recall, $T \subseteq T_{\leq\gamma} \cdot (\Sigma_{\mathtt{i}}^* + \Sigma_{\mathtt{o}}^*)^n$, and our goal is to show that there is a bound k such that it suffices to focus on constructing T_k-controlled uniformizers instead of T-controlled uniformizers, meaning that we can focus on uniformizers in which the length of output blocks is bounded by k after the lag has exceeded γ at some point.

The core of the proof is to show that if the lag between input and output becomes very large ($\gg \gamma$), it is not necessary to consider the complete input that is ahead to determine the next output block, but an abstraction (in the form of profiles) suffices. Note that if the lag has exceeded γ at some point the number of remaining output blocks is at most $\lceil n/2 \rceil$.

As a result, given an input word $x \in \Sigma_{\mathtt{i}}^*$, we are interested in the state transformation that is induced by $(x, \pi_{\mathtt{o}}(w))$ in $\mathcal{A}$ (recognizing S_{can}) and by w in $\mathcal{B}$ (recognizing T) for each word $w \in \Sigma_{\mathtt{io}}^*$ such that $|\pi_{\mathtt{o}}(w)| \leq |x|$ and $\mathit{shift}(w) \leq \lceil n/2 \rceil$. In words, we are interested in the state transformations that can be induced by x together with outputs of same or smaller length that are composed of at most $\lceil n/2 \rceil$ different output blocks.

For $x \in \Sigma_{\mathtt{i}}^*$, this kind of information is accurately represented by the set of all reduced input state transformation trees with parameters x and $\lceil n/2 \rceil$.

The same considerations with switched input and output roles apply for an output word $y \in \Sigma_{\mathtt{o}}^*$.

▶ **Definition 19** (Input profile)**.** For $x \in \Sigma_{\mathtt{i}}^*$, we define its *profile* P_x as $(\tau_x, \mathrm{STT}_x^{\lceil n/2 \rceil})$, where

$$\mathrm{STT}_x^{\lceil n/2 \rceil} = \bigcup_{(p,q) \in Q_{\mathcal{B}} \times Q_{\mathcal{A}}} \{\mathit{red}\big(\mathrm{STT}^{\lceil n/2 \rceil}(x, p, q)\big)\}.$$

Similarly, we define *output profiles*, a formal definition can be found in the full version.

A note on the number of different profiles. Profiles are based on reduced STTs with parameter $\lceil n/2 \rceil$, where n bounds $\mathit{shiftlag}(T)$. The size of the set of these STTs is non-elementary in n, hence also the number of profiles. This implies a non-elementary complexity of our decision procedure.

Furthermore, let $\mathcal{P}_{\mathtt{i}}$ be the set $\bigcup_{x \in \Sigma_{\mathtt{i}}^*} \{P_x\}$ of all input profiles and $\mathcal{P}_{\mathtt{o}}$ be the set $\bigcup_{y \in \Sigma_{\mathtt{o}}^*} \{P_y\}$ of all output profiles. For a $P \in \mathcal{P}_{\mathtt{i}} \cup \mathcal{P}_{\mathtt{o}}$, let z be a *representative* of P if z is a shortest word such that $P = P_z$.

We show that from the profiles of two words x_1 and x_2 one can compute the profile of the word $x_1 x_2$. Hence, the set of profiles can be equipped with a concatenation operation, i.e., for words x_1 and x_2 we let $P_{x_1} P_{x_2} = P_{x_1 x_2}$. We obtain the following.

▶ **Lemma 20.** *The set of input profiles is a monoid with concatenation; the set of output profiles is a monoid with concatenation.*

A word $x \in \Sigma_{\mathtt{i}}^*$ and its profile P_x are called *idempotent* if $P_x = P_{xx}$. As a consequence of Ramsey's Theorem (see e.g., [5]) we obtain the following lemma.

▶ **Lemma 21** (Consequence of Ramsey)**.** *There is a computable $r \in \mathbb{N}$ such that each word $x \in \Sigma_{\mathtt{i}}^*$ with $|x| \geq r$ contains a non-empty idempotent factor for the concatenation of profiles.*

Now, we have the right tools to prove that the existence of a T-controlled uniformizer implies that there also exists a T_k-controlled uniformizer for a computable k. For the remainder, we fix two bounds.

▶ **Assumption 22.** Assume r_1 is chosen as in Lemma 21 and r_2 is chosen as the smallest bound on the length of representatives of output profiles. Wlog, assume $r_1, r_2 > \gamma$.

Finally, we are ready to prove the key lemma, that is, Lemma 23, which shows that it is sufficient to consider uniformizers in which the length of output blocks is bounded.

Recall, a uniformizer works asynchronously, which leads to large lag. First, we show that if the output is lagged more than r_1 symbols, meaning, the input that is ahead contains an idempotent factor, it suffices to consider output blocks whose length depends on the idempotent factor. Secondly, we show that it suffices to consider uniformizers in which the output is ahead at most r_2 symbols. The combination of both results yields Lemma 23.

Recall, by Asm. 13, $T \subseteq T_{\leq\gamma} \cdot (\Sigma_{\mathtt{i}}^* + \Sigma_{\mathtt{o}}^*)^n$ and by Def. 14, $T_i = T \cap \left(T_{\leq\gamma} \cdot (\Sigma_{\mathtt{i}}^* + \Sigma_{\mathtt{o}}^{\leq i})^n\right)$ for $i \geq 0$.

▶ **Lemma 23.** *If S has a T-controlled uniformizer, then S has a T_k-controlled uniformizer for a computable $k \geq 0$.*

The proof of the above lemma yields that k can be chosen as $r_1 + r_2$. This concludes the first part of the proof of Theorem 10. For the second part, we prove that the problem whether S has a T_i-controlled uniformizer for an i reduces to the question whether $T_i(S)$ has a subset uniformizer for a suitable $T_i(S)$ as defined below in Lemma 24.

Reduction. The next lemma shows that from S a regular $T_i(S)$ can be obtained such that $T_i(S)$ consists of all T_i-controlled synchronizations w with $[\![w]\!] \in [\![S]\!]$.

▶ **Lemma 24.** *For $i \geq 0$, the language $T_i(S) = \{w \in \Sigma_{\mathtt{io}}^* \mid w \in T_i \text{ and } [\![w]\!] \in [\![S]\!]\}$ is a T_i-controlled effectively constructible regular language.*

We are ready to prove the main theorem of this paper.

Proof sketch of Theorem 10. By Lemma 23 we know that if S has a T-controlled uniformizer, then S has a T_k-controlled uniformizer for a computable $k \geq 0$. Let $T_k(S)$ be defined as in Lemma 24.

We can show that S has a T-controlled uniformizer iff $\mathrm{dom}([\![S]\!]) = \mathrm{dom}([\![T_k(S)]\!])$ and $T_k(S)$ has a subset uniformizer which is decidable by Theorem 8. ◀

5 Conclusion

In this paper we considered uniformization by subsequential transducers in which the allowed input/output behavior is specified by a regular set of synchronizations, the so-called resynchronized uniformization problem. An overview over our results can be found in Table 1.

For future work we want to study other problems of this kind, e.g., study whether the resynchronized uniformization problem is decidable for a given rational relation as source language and a given 'recognizable' target language in the sense that the target language is controlled by a synchronization language that synchronizes recognizable relations.

References

1 Jean Berstel. Transductions and context-free languages `http://www-igm.univ-mlv.fr/~berstel/`, 2009. URL: `http://www-igm.univ-mlv.fr/~berstel/`.

2 J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969. `doi:10.1090/S0002-9947-1969-0280205-0`.

3 Arnaud Carayol and Christof Löding. Uniformization in Automata Theory. In *Proceedings of the 14th Congress of Logic, Methodology and Philosophy of Science Nancy, July 19-26, 2011*, pages 153–178. London: College Publications, 2014.

4 Alonzo Church. Logic, arithmetic and automata. In *Proceedings of the International Congress of Mathematicians*, pages 23–35, 1962.

5 R. Diestel. *Graph Theory, 2nd Edition*, volume 173 of *Graduate Texts in Mathematics*. Springer, 2000.

6 Diego Figueira and Leonid Libkin. Synchronizing relations on words. *Theory Comput. Syst.*, 57(2):287–318, 2015.

7 Emmanuel Filiot, Ismaël Jecker, Christof Löding, and Sarah Winter. On equivalence and uniformisation problems for finite transducers. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 125:1–125:14, 2016. `doi:10.4230/LIPIcs.ICALP.2016.125`.

8 Christiane Frougny and Jacques Sakarovitch. Synchronized rational relations of finite and infinite words. *Theor. Comput. Sci.*, 108(1):45–82, 1993. `doi:10.1016/0304-3975(93)90230-Q`.

9 Michael Holtmann, Łukasz Kaiser, and Wolfgang Thomas. Degrees of lookahead in regular infinite games. In *Foundations of Software Science and Computational Structures*, volume 6014 of *Lecture Notes in Computer Science*, pages 252–266. Springer, 2010. `doi:/10.1007/978-3-642-12032-9_18`.

10 Frederick A. Hosch and Lawrence H. Landweber. Finite delay solutions for sequential conditions. In *ICALP*, pages 45–60, 1972.

11 J. Howard Johnson. Uniformizing rational relations for natural language applications using weighted determinization. In *Proceedings of the 15th International Conference on Implementation and Application of Automata*, CIAA'10, pages 173–180, Berlin, Heidelberg, 2011. Springer-Verlag. URL: `http://dl.acm.org/citation.cfm?id=1964285.1964304`.

12 M. Nivat. Transductions des langages de Chomsky. *Ann. de l'Inst. Fourier*, 18:339–456, 1968. in french.

13 Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

14 Wolfgang Thomas. Church's problem and a tour through automata theory. In *Pillars of Computer Science, Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday*, volume 4800 of *Lecture Notes in Computer Science*, pages 635–655. Springer, 2008.